Accepted Manuscript

Predatory Search-based Chaos Turbo Particle Swarm Optimization (PS-CTPSO): A new particle swarm optimisation algorithm for Web service combination problems

Xiaolong Xu, Hanzhong Rong, Ella Pereira, Marcello Trovati

PII: S0167-739X(17)33034-0
DOI: https://doi.org/10.1016/j.future.2018.07.002
Reference: FUTURE 4320

To appear in: *Future Generation Computer Systems*

Received date : 31 December 2017
Revised date : 18 April 2018
Accepted date : 1 July 2018

Please cite this article as: X. Xu, H. Rong, E. Pereira, M. Trovati, Predatory Search-based Chaos Turbo Particle Swarm Optimization (PS-CTPSO): A new particle swarm optimisation algorithm for Web service combination problems, *Future Generation Computer Systems* (2018), https://doi.org/10.1016/j.future.2018.07.002

Predatory Search-based Chaos Turbo Particle Swarm Optimization (PS-CTPSO): a New Particle Swarm Optimisation Algorithm for Web Service Combination Problems

Xiaolong Xu and Hanzhong Rong

Jiangsu Key Laboratory of Big Data Security Intelligent Processing
Nanjing University of Posts and Telecommunications
Nanjing, China

Ella Pereira and Marcello Trovati

Department of Computer Science
Edge Hill University
Ormskirk, UK

Abstract

Web service combinatorial optimisation is an NP problem (that is, characterised by a nondeterministic polynomial time solution), based on the logical relationship between each service pair. As a consequence, obtaining the best Web service composition scheme is typically a complex task. In this article, we propose the the Predatory Search-based Chaos Turbo Particle Swarm Optimization (PS-CTPSO) algorithm, a chaotic particle swarm optimisation algorithm based on the predatory search strategy, which has significant potential to enhance the overall performance of the Autonomous Cloud.

This is achieved by integrating a predatory search and cotangent sequence strategies with the particle swarm optimisation algorithm. More specifically, the PS-CTPSO algorithm identifies a feasible service via a global search, and subsequently, it obtains suitable candidate services within the corresponding chain. The different Web services are grouped into the same class, depending on whether they have the same input and output sets, thus reducing the number of combinations and improving the searching efficiency. In the initialisation phase, the PS-CTPSO component introduces the cotangent method, rather than a ran-

dom one, which defines individual candidate services within the corresponding classes, creating a feasible service chain. In the update phase, a novel set of rules is used to perturb the velocities and positions of particles for assessing the ideal global search capabilities and adaptability. This effectively overcomes any premature problem, which commonly occurs in traditional PSO (Particle Swarm Optimisation) algorithms, and logic optimisation ensures the diversity of the final combination scheme. In this article, a prototype system (BestWS) is created, based on the directed graph generated by the logic relationships between Web services and the PS-CTPSO, Graph-Based Particle Swarm Optimization (GB-PSO), Chaos Particle Swarm Optimisation (CS-PSO) and Chaos Particle Swarm Optimization with Predatory Search strategy (PS-CSPSO) algorithms. The experimental results demonstrate that the cotangent sequence is more suitable than the chaotic one in the field of Web service combination optimisation. Furthermore, compared with the typical implementation of GB-PSO and PS-CSPSO, PS-CTPSO obtains better results, whilst attaining the global optimum with fewer iterations, and with an improved overall ergodicity.

Keywords: Autonomous Cloud, Particle Swarm Optimisation, Predatory Search Strategy, Chaos Search, Web Combination Optimisation

1. Introduction

Web services are crucial for any Cloud Computing Services platform, and have been investigated within many fields, with particular emphasis to the Quality of Service (QoS) task, which is a crucial factor for the success of Web-service applications. In fact, the applications of QoS to Web service tasks have drawn considerable attention from the research community. More specifically, the identification of the most suitable service whilst performing combinatorial optimisation to address QoS, is essential in Web service composition.

Furthermore, there are other important applications. In [1], the authors specifically address the fact that businesses must fully exploit unstructured information that is unstructured and with a much stronger social connotation. In

particular, they focus on company's contact Web forms using the textual analytics, processing of frequently asked questions and a rule-based system. In [2], an extension to Media Independent Handover Services, which allow the exchange of service information along with network information at handover. This constitutes the core of a quality-driven service discovery, which is capable of discovering services, whilst facilitating their mutual handover.

An increasing number of computing and Web services are being designed within autonomous cloud systems, raising new opportunities as well as challenges. In particular, the complexity of the cloud infrastructure is rapidly increasing to address such issues, which require the design, assessment and investigation of novel approaches to enhance the overall efficiency [3, 4]. As a consequence, the integration of specific optimisation algorithms with Web services is likely to lead to to an improved performance [5].

Intelligent optimisation algorithms are used to assess the combinatorial optimisation problem of Web service. In [6, 7, 8], Particle Swarm Optimisation (PSO) is introduced, which is inspired by the predatory behaviour of birds, also defined as a random search algorithm [9]. It efficiently identifies the optimal position by using swarm intelligence due to a smaller number of parameters, as well as exhibiting fast convergence, simple modelling and easy implementation. However, there is still scope for improvement of its performance. In fact, PSO algorithm can easily fall into a local optimum during the process of calculation and therefore, lead to a premature solution while solving multimodal optimisation problems, due to its fast convergence speed. More specifically, it randomly generates a group of particles as a stochastic solution, by using a random search strategy in the initialisation phase. This allows an efficient identification of the optimal solution through iterations, via an iterative process based on the position and velocity of particles. However, its intrinsic randomness, may lead to a lack of its overall efficiency.

The main approach to improve the PSO algorithm, focuses on the optimisation of its performance by dynamically adjusting the search step length, as well

as optimising the particle update strategy [10, 11]. Another approach involves the combination of the PSO algorithm with other intelligent optimisation algorithms, such as genetic algorithm, ant colony algorithm, simulated annealing algorithm, etc. [12, 13, 14]. However, the majority of the current research focuses on the study of continuous optimisation problems, as discussed in [15, 16, 17].

In this article, an improvement of the PSO algorithm is proposed, which is based on the combinatorial optimisation of Web service tasks. In particular, this is very significant to the overall performance of Autonomous Cloud.

The global search and local search have been adjusted by introducing a predatory search strategy, and specific aspects of chaos theory are applied to the initialisation and updating of the particle swarm algorithm. Finally, a Web service oriented chaos particle swarm optimisation algorithm PS-CTPSO, which is based on predatory search strategy.

In particular, a predatory search (PS) strategy is introduced within PS-CTPSO to search for a viable service chain, whilst strengthening the local search for particles associated with feasible solution, which improves the accuracy of the algorithm.

Web services are subsequently classified via specific properties, such as similar input and output sets, which may lead to a combination reduction and thus allowing the complete enumeration of all combination schemes.

A cotangent sequence is also introduced in the initialisation phase of the particles position. The corresponding properties of ergodicity, regularity and pseudo-randomness of the cotangent sequence define the random searching strategy of PSO algorithm, which enhances the global searching ability of PS-CTPSO algorithm.

A prototype system (BestWS) was built to test and evaluate the proposed approach, based on the logic relationships between different Web services. The rest of the paper is organised as follows. Section 2 discusses the related work, and Section 3 introduces PS-CTPSO in details. The evaluation and experimental analysis are discussed in Section 4, and finally, Section 5 concludes the work

by summarising the main contributions and commenting on future directions of our work.

2. Related Work

PSO is an optimisation meta-heuristic inspired by the foraging behaviour of birds in nature, based on the following scenario. A flock of birds are randomly distributed over an area with only one piece of food, represented by the dot in Figure 1(a). Even though no bird knows where the food is, it is, however, aware of its distance from the food. Furthermore, the bird nearest to the food can notify other birds to fly to it. If the position of the food is assumed to be at as shown in Figure 1(b), each bird can be seen as a particle, and the distance between a bird and the food as the objective function. As a consequence, the bird flocks foraging process can be regarded as a function optimisation process. In Figure 1(b), X_i is the current global optimal particle (that is the closest particle to the goal), and its distance from goal $Nbest_i$ is the global optimal value [6, 7].

In PSO, every set of particles is defined by the position and velocity of its elements. However, the process of searching for the global optimum is typically an NP-hard problem [20]. The particles iteratively update their positions according to their individual local optimal position and the global optimal position visited so far. More specifically, the new position of a particle i is defined as:

$$X_i(t+1) = X_i(t) + V_i(t+1), \tag{1}$$

where t is the current status, $X_i(t)$ is the current position of the particle, and $V_i(t+1)$ is its new velocity. Note that, each time difference is defined as a time unit. The velocity of particle i is then defined as

$$V_i(t+1) = wV_i(t) + c_1r_1(X_i^p - X_i(t)) + c_2r_2(X_i^g - X_i(t)), \tag{2}$$

where

- $V_i(t)$ is the current velocity of the particle

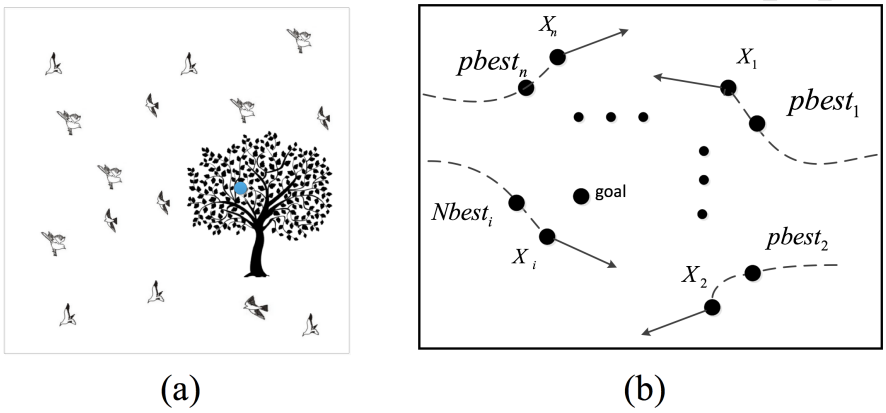


Figure 1: Simulation of the bird flocks foraging.

- X_i^p is the best position so far visited by the particle (i.e. the local best position)
- X_i^g is the global best position so far visited by a particle at the swarm level
- w, c_1 and c_2 are constants associated with the significance of each component of the velocity, and
- r_1, r_2 are the random values within the interval $[0, 1]$.

In recent years, much research has been carried out on Web service combination problems, especially using the improved PSO algorithm. In [18], the authors propose an enhanced dynamic particle swarm optimisation (IDPSO) algorithm based on QoS to address Web combination optimisation. This is based on the process of selecting larger step size values when particles are far from the optimum target, and smaller ones when particles tend towards an optimum target. Thus, the IDPSO algorithm constantly adjusts the position of particles position by altering the step size values, in order to obtain a better performance. However, when the optimum and global optimal targets do not coincide, the particle will easily approximate local optimum [18].

In [19], the authors propose an improved particle swarm optimisation based on di-graphs (GB-PSO) for solving Web combination optimisation, in which all service nodes are traversed. Subsequently, all service chains satisfying the requirements of the initial input set and the object output set are identified, which generate digraphs used to identify the optimal service chain. However, when the number of service nodes is large, the efficiency of the algorithm is very low, which is likely to lead to some invalid services [19].

In [20], a dynamic and discrete particle swarm optimisation (DDPSO) for addressing dynamic Web service combination optimisation problems is discussed. In order to reduce the service selection time and space costs, the authors introduce a method to eliminate redundant candidate services, whilst addressing any premature convergence issues. Eventually, the whole particles are divided into z classes, where each of them only retains a single element, and all other redundant candidate services are eliminated, thus leaving a service chain. However, due to adopting the random strategy in the initialisation and update phase, it is not suitable for large numbers of candidate services [20]. In [21], the authors discuss a modified particle swarm optimisation algorithm based on sub-particle circular orbit, and zero-value inertial weight (MDPSO). MDPSO utilises trigonometric-function-based, non-linear dynamic learning factors, and a prediction method of population premature convergence, which provide a better balance between the local exploring ability and the global converging ability of particles [21]. MDPSO is mainly suitable for solving the composition optimisation problem of Web Service, and it does not consider the logical relationships between services. Therefore, it is not suitable for application to the field of the combinatorial optimisation with logical relations.

3. Description of the PS-CTPSO Algorithm

The particle swarm algorithm needs to address the following aspects:

1. The exponential growth of the complexity of the PSO algorithm;

2. During the process of searching for the optimal solution, random search strategy cannot guarantee the diversity of the final solution;
3. Its universality is poor, and finally;
4. The PSO algorithm needs to consider the logical relationships between Web services.

In order to solving the above problems, this article introduces PS-CTPSO algorithm, which can be divided into the following steps:

Step 1: a predatory search strategy is defined to obtain a feasible service chain in the whole process. In other words, it searches for a combination of user services to meet the requirements of the Web service chain.

Step 2: the optimal candidate service obtained from Step 1 is identified. During this stage, Web services with the same input and output sets are grouped into the same service class, so that the number of combinations is reduced, and the search efficiency is improved.

Step 3: during the initialisation of the particle swarm, a chaotic search method is utilised instead of random initialisation. Subsequently, a chaotic sequence is used to take a candidate service in order to create a feasible service chain.

Step 4: in the update phase, a new chaotic perturbation rule is defined to replace the random searching rule. The velocity and position of the particles are perturbed, which allows the algorithm to have good global search ability and adaptability. This, in turn, can effectively address the issue of any premature convergence problem, whilst ensuring the diversity of the final combination of programs.

Step 5: finally, either a global optimal solution based on the current service chain, or the maximum number of iterations, is reached. If the latter occurs, a different global search strategy is implemented to identify a feasible service chain. This is based on performing a particle swarm implementation to obtain a global optimal solution.

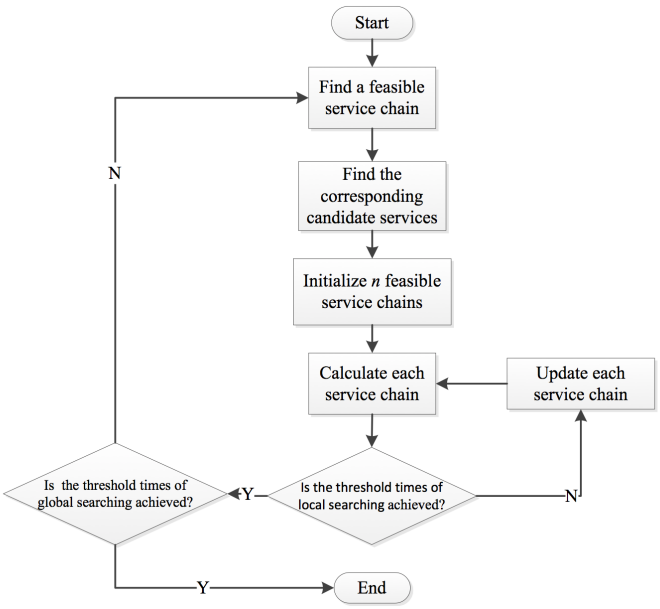


Figure 2: The flow chart of PS-CTPSO algorithm.

Figure 2 depicts the PS-CTPSO algorithm flow chart.

3.1. Model of Web Service Combination Optimisation

Definition 1. Let $W = \{W_1, \ldots, W_m\}$ be the set related to the same service function, defined by identical input and output values, where W_i refers to the $i-$th class service.

Definition 2. Let $w_{i,j}$ be the basic logical unit, which defines the service composition. The $i-$th service class candidate service is defined by $W_i = \{w_{i,1}, \ldots, w_{i,k}\}$. Its service class corresponding to the input set is $I_{w_i} = I_{w_{i,1}} = \cdots = I_{w_{i,k}}$, and the output set is $O_{w_i} = O_{w_{i,1}} = \cdots = O_{w_{i,k}}$.

Each candidate service contains a QoS vector, defined by the following four parameters [22]:

Response Time t: the execution time t of a service is equal to the time interval between the point at which the request is sent and when the result is received.

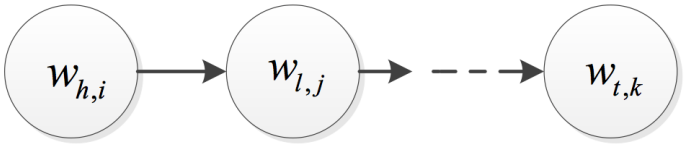


Figure 3: Web services combination logical structure diagram.

Execution Cost c**:** this refers to the execution cost c of a service.

Reliability r**:** this is the probability of a correct response within the maximum expected time, which is expressed as

$$r = \frac{N_s}{N_t}, \tag{3}$$

where N_s indicates the times which is successfully called during the observation time, and N_t indicating the total times which is invoked during the observation time.

Availability a**:** this is the probability of a service being used, which is defined as:

$$a = \frac{T_\lambda}{\lambda}, \tag{4}$$

where λ is the constant value, which depends on the type of service, and T_λ is the available time of service within λ time.

According to Definitions 1 and 2, the service composition consists of candidate services from different service classes, described as $W_c = \{0, 1, \ldots, 0, \ldots, 1\}$. In particular, $w_{i,j} = 1$ if and only if the corresponding elements in the service chain are joined. This depends on the logical order depicted as a directed graph, as shown in Figure 3.

The Web service combination optimisation model consists of the following steps:

1. Assume that there are m service classes, namely $W = \{W_0, \ldots, W_{m-1}\}$.
2. For every service class $W_i = \{w_{i,0}, \ldots, w_{i,k_{m-1}}\}$, there are a number of candidate services, where k_i refers to the $i-th$ service class. Furthermore, each service has an attribute parameter that describes its quality.

3. Suppose there is a service chain $S_c = s_0 \rightarrow s_1 \rightarrow \cdots \rightarrow s_{m-1}$ (s_i is equivalent to $w_{i,j}$).

4. In the proposed model, the functions $getInput()$ and $getOutput()$ are defined to obtain the set of input and output for the corresponding service. As a consequnce, the quality of service must be maximised. In other words, the following needs to be evaluated

$$\max\left(\alpha R + \beta A + \gamma\frac{1}{T} + \mu\frac{1}{C}\right). \tag{5}$$

such that $getOutput(s_i) = getInput(s_{i+1})$, for $s_i \in S_c$, and $\alpha+\beta+\gamma+\mu = 1$, which are the weights attributes of QoS. Furthermore,

$$R = \prod_{i=1}^{m} r_i, \quad T = \sum_{i=1}^{m} t_i, \quad C = \sum_{i=1}^{m} c_i, \quad A = \prod_{i=1}^{m} a_i,$$

where R, T, C and A respectively refer to the product (or sum) of each QoS properties of service chain. More specifically, R and A are the products of each service reliability and availability, respectively. Similarly, T and C are the sum of each service response time and execution time, respectively.

The PS-CTPSO algorithm is divided into three steps:

1. Find a feasible service by global searching, as well as the candidate service of the service chain;
2. Initialisation of particles via the cotangent sequence, and
3. Calculation of the fitness function value of each particle to evaluate the suitability of each particle, whilst selecting the optimal particle to update the other particles.

The detailed steps as described in Sections 3.2 – 3.5.

3.2. Searching for a Feasible Service Chain

As discussed above, in order to obtain a feasible service chain, it is crucial to consider the Web service combination, the logical relationships between services, and the QoS quality of services. However, when the service dataset is

very large, identifying the global optimal solution might not be computationally possible. Motivated by the properties of the Web service composition optimisation problem, this article proposes a complex combination optimisation scheme, which can be used to solve combination optimisation problem with a logical relationship.

The predatory search is a spatial searching strategy proposed by Alexandre Linhares in 1998 [23] to solve the combinatorial optimisation problem, based on food searching strategies within the animal realm. At first, the whole space must be searched to identify a feasible solution within local spaces. Since most of the searching time is spent within a local space near the optimal solution, the predatory search strategy is efficient and fast. As a consequence, when it is combined with the particle swarm algorithm, the overall efficiency is improved, leading to a more accurate identification of the solution.

Therefore, the predatory search algorithm is based on a balance between local and global searches. On the one hand, the global searching process is responsible for exploring the breadth of the solution space and improving the quality of the searching, whilst avoiding any local optimal solution. On the other hand, the local search component aims to develop the depth of a better solution space, and it exhibits fast searching time [24].

Chaos search, which is applied to local search, is associated with itineraries exhibiting pseudo-randomness, ergodicity and regularity properties. Based on the features of Web services composition optimisation process, searching a service chain has clear dynamical properties. In this article, a cotangent sequence is utilised to generate a pseudo-random sequence. Due to the range of the cotangent sequence being $(-\infty, +\infty)$, the traversal range is wider, which results to a more adaptable application environment to the field of Web service combination. The cotangent sequence is defined as

$$a_{n+1} = \cot(a_n), \tag{6}$$

where a_n is a value of cotangent sequence for the $n-$th iteration. Furthermore, for some specific initial values, Equation 6 can generate a pseudo-random se-

quence α_i, for $i \geq 0$.

In order to integrate the predatory search strategy with the particle swarm algorithm, local and global thresholds must be defined. These are used to ensure whether a suitable solution can be found, by selecting either a global or local search.

Consider the scenario where all the data related to a Web service is stored as a text file, and each line consists of four parts corresponding to the information related to such a Web service. For example, if we consider a weather forecast service, the first component may be the service name; the second one may be the input set (with comma separated input attribute values); the third part may be the output set; and finally, the attribute values of QoS (including the response time, the execution cost, availability and reliability) would be the fourth component.

In the searching process, in order to ensure any invalid service is avoided, which is equivalent to no effect on the final service chain if this is removed, a search set is added. This is defined by the variable *searchSet* in Algorithm 1, which is used to store the input set of the service chain.

The detailed steps are as follows:

- Prior to a Web service joining the service chain, the input set of the Web service must be associated with *searchSet*;
- If it does not exist, and satisfies the requirements, the Web service can be added to the service chain. Otherwise, it cannot join in the service chain.
- If the iteration searching is complete, whilst not identifying a feasible service chain, the *searchSet* is re-initialised Subsequently, in the next iteration the Web services will not be searched, until either a feasible service chain is identified, or the maximum number of iterations is reached.

Algorithm 1 formalises the above steps, where N represents a total number of Web services, and the variable tag is used to record the last service which is added to the service chain.

Algorithm 1 Discovering a feasible service chain

```
1: Input: I, O
2: Output: S_c
3: serviceList = readFile("filePath")
4: chosen[N] = false {false means that the service without being searched, true means the service has been searched}
5: objectOutputSet = I {Initialise the target output set}
6: searchSet = null {store the input set of the service chain}
7: tag = 0
8: hashMap < NodeBean, ArrayList < NodeBean >> () = null {store relationship between Web services}
9: for i = 0 to N − 1 do
10:    nodeBean[i] = getBean(serviceList) {the function get-Bean() is used to obtain a service}
11: end for
12: while True do
13:    for j = 0 to N − 1 do
14:       for i = 0 to N − 1 do
15:          if !chosen[i] and !searchSet.containsAll(nodeBean[i].input and objectOutputSet.containsAll(nodeBean[i].input)) then
16:             if !objectOutputSet.output.containsAll(nodeBean[i].output)) then
17:                addEdge(hashMap, nodeBean[tag], nodeBean[i])
18:                searchSet.addAll(nodeBean[i].input)
19:                tag = i
20:                chosen[i] = true
21:                objectOutputSet.addAll(nodeBean[i].output)
22:                if objectOutputSet.containsAll(O) then
23:                   break
24:                end if
25:             end if
26:          end if
27:       end for
28:       if objectOutputSet.containsAll(O) then
29:          addEdge(hashMap, nodeBean[tag], O)
30:          break
31:       end if
32:    end for
33:    if !objectOutput.containsAll(O) then
34:       searchSet.clear() {clear the set of variable searchSet}
35:    else
36:       S_c = generateServiceList(hashMap) {Generate the service chain, and exit the while loop}
37:       break;
38:    end if
39: end while
40: return S_c
```

While Algorithm 1 finds a feasible service chain S_c, Algorithm 2 identifies suitable candidate services of the service chain, based on the output of Algorithm 1.

Algorithm 2 Discovering a relevant services for composition

Input: S_c {search to a viable service chain}
Output: $List < int[] >$ COMMENTstore int type of the array List
$List < int[] > list =$ null
for $k = 0$ to $Sc.size()$ **do**
  for $j = 0$ to $nodeBean.length$ **do**
    $int[]commonService = 0$
    $i = 0$
    if $nodeBean[getNodeBeanIndex(k)].input.equals(nodeBean[j].input)$ **and** $nodeBean[getNodeBeanIndex(k)].output.equals(NodeBean[j].output)$ **then**
      $commonService[i++] = j$
    end if
  end for
  $list.add(commService)$
end for
return $list$

In particular, in Algorithm 2, *list* is a candidate service array for storing each service on the corresponding service chain, and the *commonService* array is a candidate service array.

3.3. Chaos Initialisation Process

The chaos initialisation process refers to the sequence defined by the cotangent equation, which randomly identifies the particle initial value. The parameters of chaos initialisation are as follows:

1. The service chain is assumed to consist of m services.

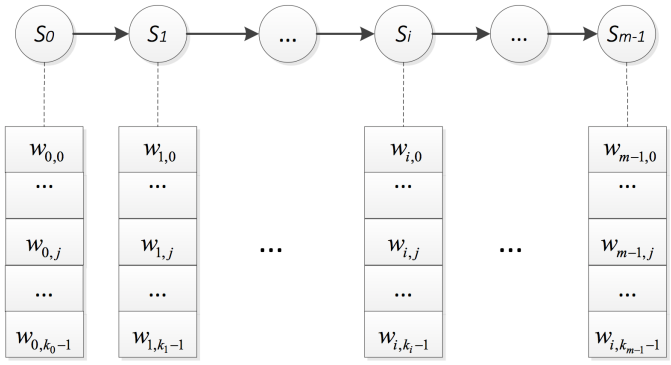


Figure 4: The service chain and the corresponding candidate services.

2. Each element in the service chain contains a category of candidate services, which are divided into m categories, defined as $W_i = (w_{i,0}, w_{i,1}, \ldots)$, for $i = 0, \ldots, m-1$. The service chain and the corresponding candidate service are shown in Figure 4.
3. Via a classification process, similar services are grouped into the same category, thus reducing the number of combinations. If the services have the same input set and output set, they will be grouped into the same category. This will enable the enumeration of all combination schemes and improve the search efficiency.

Within a Web service composition optimisation problem, the optimal fitness value is not known in advance. Therefore, in this article we propose a dynamical cotangent sequence method to initialise the particles. More specifically, the main steps are as follows:

1. Let the number of Web services in the service chain be m, and assume that the number of each candidate services is k_i.
2. Randomly select m values in the interval $[0, \pi]$ random space, so that the initial values of the sequence are $\phi_{0,0}, \ldots, \phi_{m-1,0}$
3. By iterating $a_{n+1} = \cot(a_n)$ the cotangent sequence values are calculated.
4. The number candidate services in each category determines the number

of decimal points. In particular, if the $i-$th category has 60 elements, two decimal points are evaluated modulo 60, so it will generate the value between 0 and 59, which corresponds to a certain candidate service of this category.

The detailed steps of chaos initialisation are as follows:

- As discussed above, let the length of service chain be m, and let each service in the service chain have k_i candidate services.
- Consider the $i-$th particle, and assume that $\phi_i = \log(k_i)$ for the cotangent sequence value $\phi_{0,i}$ of the corresponding service, intersecting a ϕ_i digit value as an integer values $u_{0,i}$, in the range $[0, I_i)$.
- Subsequently, by evaluating $u_{0,i} \bmod k_i$, define a chaos value between $\xi_{0,i} \in [0, k_i)$, which corresponds to a candidate service in the $i-$th category, so that the corresponding particle position is equal to 1, while the others are set to 0.
- Similarly, the same operation is performed on the other services.

For n particles, the above steps must be repeated n times, which defines an $n \times m$ cotangent sequence matrix C, as shown below

$$C = \begin{bmatrix} \phi_{0,0} & \cdots & \phi_{0,m-1} \\ \vdots & \cdots & \vdots \\ \phi_{n-1,0} & \cdots & \phi_{n-1,m-1} \end{bmatrix} \overset{\Gamma(\phi_{i,j}) \mod k_i}{\longrightarrow} \begin{bmatrix} \xi_{0,0} & \cdots & \xi_{0,m-1} \\ \vdots & \cdots & \vdots \\ \xi_{n-1,0} & \cdots & \xi_{n-1,m-1} \end{bmatrix} \tag{7}$$

where $\Gamma(\phi_{i,j}) \mod k_i$ defines the (chaotic) value for the input set $O_i = \lceil \log(k_i) \rceil$. In the initialisation phase, the velocity V_i and the local best position of particle i are all equal to X_i, that is

$$X_i = V_i = P_i \qquad \text{for } i = 1, \ldots, n-1 \tag{8}$$

3.4. Fitness Function

In this article, the service combination includes four QoS parameters, namely reliability, response time, execution cost and availability. The values of reliability and availability, are between 0 and 1, so the parameters of the response

time and the execution cost must be normalised. In this context, we define the maximum value of the response time R_{max} and the maximum value of the execution cost C_{max} in all candidate services, which are used to normalise the execution and time costs as follows:

$$\begin{aligned} t_{i_s} = \frac{t_i}{T_{max}} \qquad c_{i_s} = \frac{c_i}{C_{max}} \\ f_i = \alpha r_i + \beta a_i + \gamma \frac{1}{t_{i_s}} + \mu \frac{1}{c_{i_s}} \\ F = \sum_{i=0}^{m} f_i \end{aligned} \tag{9}$$

Recall that $\alpha + \beta + \gamma + \mu = 1$, which correspond to the weights of the QoS attributes.

3.5. Chaos Perturbation

During the updating process of particles, their velocities and positions are perturbed, so that the search space will be sufficiently traversed.

The main purpose of the chaos perturbation is to prevent the particle swarm algorithm from converging to the local optimal value. In fact, if this happens, in the next iteration the velocity and position of particles are perturbed, so that particles can move away from the local optimal value. This ensuresthat the improved algorithm can traverse a broader search space.

Via the chaos initialisation, the positions of n particles are obtained, and their corresponding fitness values are set to 0. Note that, high fitness values imply better particle positions. Subsequently, the values of f_i, P_i (for $i = 0, \ldots, n-1$), F and G are obtained. Define two velocities vectors as $V_i^p = X_i^p - X_i$ and $V_i^g = X_i^g - X_i$. As a consequence, the new velocity of the particle i is updated as

$$V_i(t+1) = wV_i(t) + c_1 r_1 V_i^p + c_2 r_2 V_i^g \tag{10}$$

In particular, the difference between two positions X_i^p and X_i is evaluated as

$$X_i^p - X_i = (v_{i,0}^p, \ldots, v_{i,j}^p, \ldots), \tag{11}$$

where

$$v_{i,j}^{p} = \begin{cases} Rand(1) \text{ if } x_{i,j}^{p} = x_{i,j} \\ 0 \text{ otherwise} \end{cases} \tag{12}$$

where $Rand(1) \in \{0, 1\}$ randomly generates either 0 or 1. According to the chaos initialisation process, only a small number of items are selected in every category, as most variables are equal to 0. The updating rule of velocity is re-defined as:

$$v_{i,j}^{p}(t) = \begin{cases} v_{i,j}(t) \text{ if } v_{i,j}(t) = v_{i,j}^{p}(t) \\ -1 \text{ otherwise} \end{cases} \tag{13}$$

The addition between a position $X_i(t)$ and a velocity $V_i(t+1)$ is also re-defined as:

$$X_i(t+1) = X_i(t) + V_i(t+1) = (x_{i,0}(t), \ldots, x_{i,j}(t), \ldots) \tag{14}$$

As a consequence, the particle location update rules are as follows

$$x_{i,j}(t+1) = \begin{cases} x_{i,j}, \text{ if } v_{i,j} = 1 \\ C(x_{i,j}), \text{ if } v_{i,j}(t+1) = 0 \\ J(x_{i,j}^{p}), \text{ if } v_{i,j}(t+1) = -1 \text{ and } v_{i,j} \neq v_{i,j}^{g} \text{ and } v_{i,j} = v_{i,j}^{p} \\ J(x_{i,j}^{p}), \text{ if } v_{i,j}(t+1) = -1 \text{ and } v_{i,j} = v_{i,j}^{g} \text{ and } v_{i,j} \neq v_{i,j}^{p} \\ J(x_{i,j}^{p}), \text{ if } v_{i,j}(t+1) = -1 \text{ and } v_{i,j} = v_{i,j}^{p} \text{ and } v_{i,j} \neq v_{i,j}^{g} \\ C(x_{i,j}), \text{ if } v_{i,j}(t+1) = -1 \text{ and } v_{i,j} \neq v_{i,j}^{p} \text{ and } v_{i,j} \neq v_{i,j}^{g} \end{cases} \tag{15}$$

In (15), $C(x_{i,j})$ is the chaos perturbation function. Therefore, this will perturb the location of the particle, when the speed $v_{i,j}(t+1) = 0$ or $v_{i,j}(t+1) = -1$ and $v_{i,j}(t) \neq v^{p}(t), v_{i,j}(t) \neq v^{g}(t)$. This is equivalent to the case when their corresponding speed is 0 or -1, and the values of three variables $v_{i,j}(t+1), v_{i,j}(t)$ and $v^{g}(t)$ are different.

To illustrate this, assume that $v_{i,j}(t+1) = 0$ and $x_{i,j}$ will be perturbed.

First of all, based on the variable j, the corresponding service can be assessed via the condition that if $\sum_{i=0}^{h} k_i \leq i < \sum_{i=0}^{h+1} k_i$, then $x_{i,j}$ belongs to the h category. Subsequently, the corresponding cotangent sequence $a_{i,h}$ can be found via

$a_{n+1} = \cot(a_n)$. The new cotangent sequence value $\phi_{i,h}^{(1)}$ is then obtained at the next iteration, corresponding to the $h-$th category candidate service, that is $W_h = \{w_{h,0}, \ldots w_{h,k-1}\}$. Subsequently, $\xi_{i,h}^{(1)}$ is evaluated, as discussed above. Finally, the formula $p = j - \sum k_i$ identifies which dimension of the candidate services $x_{i,j}$ belongs to. If the chaotic value $\xi_{i,h}^{(1)}$ is equals to the value of p and $x_{i,j} = 0$, the variable $x_{i,j}$ is changed to 1, in order to ensure each category has only one selected service. All other chaos perturbing rules as shown in the following equation

$$C(x_{i,j}) = \begin{cases} 1, \text{ if } p = \xi_{i,h}^{(1)} \text{ and } x_{i,j} = 0 \\ 0, x_{i,e} = 1, \text{ for } e = \Gamma(\phi_{i,h}^{(1)}) \mod k_i \text{ if } p = \xi_{i,h}^{(1)} \text{ and } x_{i,j} = 1 \\ 0, x_{i,z} = 1, \text{ for } z = \xi_{i,h}^{(1)} + \sum_{i=0}^{h-1} k_i, \text{ if } p \neq \xi_{i,h}^{(1)} \text{ and } x_{i,j} = 1 \\ 1, x_{i,z} = 1, \text{ for } z = \xi_{i,h}^{(1)} + \sum_{i=0}^{h-1} k_i, \text{ if } p \neq \xi_{i,h}^{(1)} \text{ and } x_{i,j} = 0 \text{ and } x_{i,z} = 1 \\ x_{i,j} = 1, \text{ if } p \neq \xi_{i,h}^{(1)} \text{ and } x_{i,j} = 0 \text{ and } x_{i,z} = 0 \end{cases} \tag{16}$$

Based on the above steps, Algorithm 3 formally describes the PS-CTPSO algorithm, where:

- N is the number of particles,
- M is the total number of categories of candidate services,
- $K[i]$ is the number of services of each category,
- C is the matrix of chaotic variable,
- $X[i]$ is the position of the particle,
- $W[i][j]$ is the $j-$th service of the i category, and
- $Q[i][j]$ is the attribute value of each service.

The output of Algorithm 3 is G, which is the global best position, so the corresponding service chain is the global best service chain.

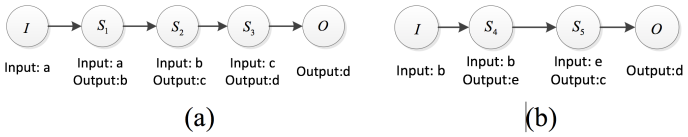


Figure 5: Replacement of service chain.

3.6. Logic Optimisation

In order to further improve the diversity of the combination scheme, we optimise the logical structure of the optimal service chain after obtaining the global optimal service chain via the PS-CTPSO algorithm. Since a service can be replaced by a service chain, consider that the optimal one is depicted as in Figure 5-(a). The service S_2 can be replaced by the service chain consisting of S_4 and S_5, as shown in Figure 5-(b). Therefore, more service chains can be identified, which satisfy the requirements. This can further improve the diversity of the combination scheme.

The detailed steps of logic optimisation are as follows:

1. Once the search is complete, the current optimal service chain is obtained. Assume that the service chain length is l, and define an integer $x \in [0, l-1]$.
2. PS-CTPSO is utilised to determine whether there is a service chain, which is equivalent to the service S_x.
3. If the equivalent service chain is found, its service chain must be assessed. If this is better than that of the former, then it must be replace it.
4. In order to improve the efficiency of the algorithm, different random numbers are generated, multiple substitutions are performed, and the number of substitutions is equal to $|\sqrt{l}|$.

3.7. Prototype System

In order to evaluate and assess the algorithms proposed in this article, a prototype system *Best Web Service Combination Recommendation System* (BestWS) was designed. The system is divided into three modules: reading

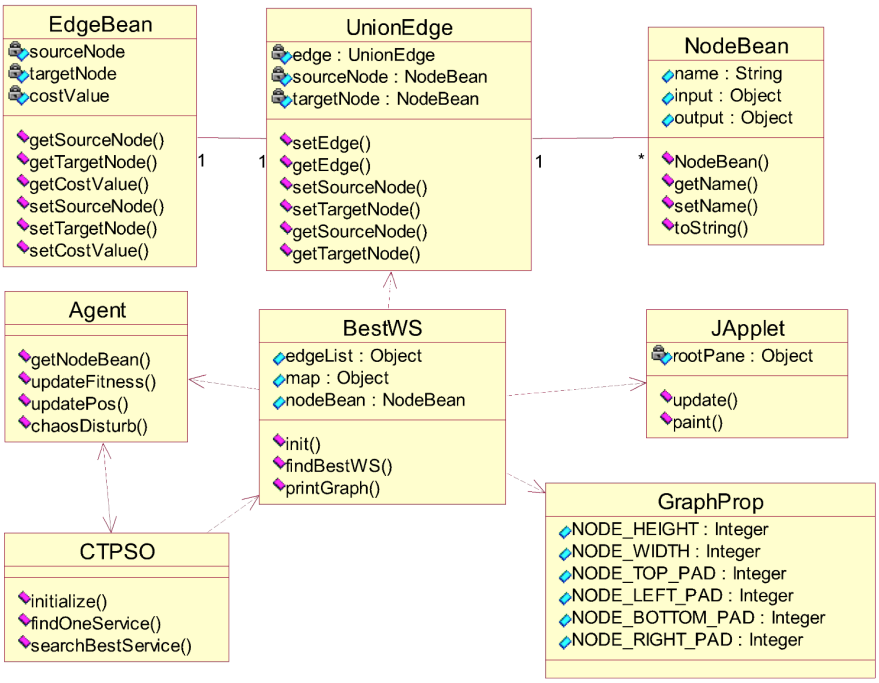


Figure 6: Web service composition system class diagram.

data, analysing algorithm and generating Web service composition logic structure diagram, as shown in Figure 6.

Firstly, the reading data module belongs to the class BestWS, and the function *init*() is used to complete the operation. Subsequently, after reading the data, BestWS schedules the class *Agent* according to the user Web service composition requirements, and it is used to control the selection of global search or local search. In particular, the global search is carried out in the class *Agent*, while the local one by the chaos search algorithm PS-CTPSO, which will analyse and calculate the optimal Web service composition. Finally, the third module generates a directed graph, and the results calculated by the algorithm are fed back to the class BestWS, and the function *paint* in JApplet will display the Web service composition logic structure.

4. Evaluation

4.1. Experimental Data and Performance Indicators

In this section, we compare the PS-CTPSO algorithm, the GB-PSO algorithm (proposed in [19]), and the improved PS-CSPSO algorithm (based on CS-PSO [25]) on the same Web Service datasets. More specifically, the order of magnitudes of dataset was defined as $30, 90, 180, 450$ and 1350, respectively. Note that, the times of iteration to identify the global optimal value, the time consumption of finding the optimal value and the diversity of the combination, are all essential aspects to consider.

4.2. Comparison of Chaotic sequence

The dynamic cotangent, random and logic chaotic sequences were assessed as shown in Table 1.

Table 1: Comparison of the ergodicity properties of the three methods.

Ergodicity coverage	1	2	3	4	5	Avg
Cotangent sequence	73%	74%	73%	73%	72%	73%
Random sequence	57%	59%	62%	58%	59%	59%
Chaotic sequence	55%	60%	66%	59%	61%	60.2%

As can be seen from the experimental results, the ergodicity of cotangent sequence is more accurate, which is associated with a chaotic behaviour, along with the logic chaotic sequence. The former is generated by the iterative cotangent function, while the latter is generated by a linear function. Although the time consumption calculated by the cotangent function is slightly larger, it has a better ergodicity and it exhibits a dynamical behaviour, so it is more suitable for Web service composition optimisation.

4.3. Comparison between CS-PSO and PS-CSPSO

In [25], a chaotic particle swarm optimisation (CS-PSO) algorithm for unordered combinatorial optimisation is proposed. This algorithm has good performance in solving unordered combination optimal problem. However, the

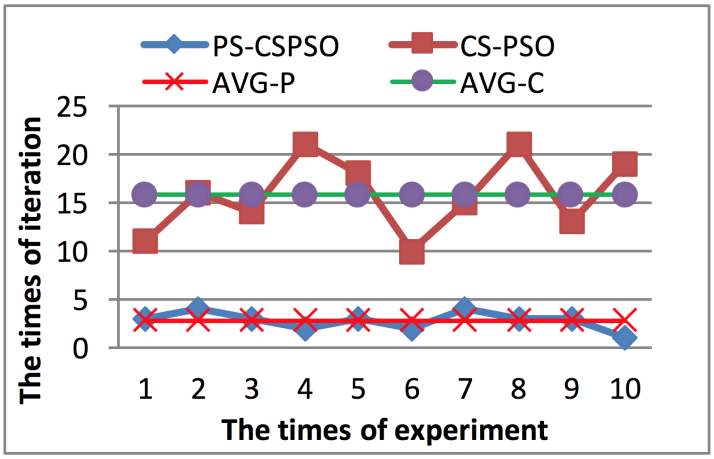


Figure 7: Comparison of iteration times in CS-PSO and PS-CSPSO.

performance of the algorithm has not been fully investigated in the ordered combination optimal problem.

Figure 7 depicts the comparison between CS-PSO and PS-CSPSO, where AVG-P and are the average number of iterations of the PS-CSPSO and CS-PSO algorithms, respectively. As can be seen, the improved CS-PSO algorithm has a better performance. Recall that the optimal rate is the rate of the optimal solutions found by an algorithm. PS-CSPSO has the optimal rate of 90%, compared to 60% of the CS-PSO algorithm.

4.4. Times of Iteration

The effective rate value of iteration is defined as the rate of the number of effective iterations with respect to the number of total iterations. If a better Web service combination is not found after one iteration, then this is recorded as invalid. Approximately, 450 data sets are used in the number of iterations are assessed, based on whether the optimal value and effective specific value of iteration are attained. Figures 8 and 9 depict the result of this part of the evaluation. More specifically, Figure 9-(a) compares results of the times of iteration, where AVG-T is the average times of iteration of PS-CTPSO algorithm, and AVG-P is the average times of iteration of PS-CSPSO. The PS-CTPSO

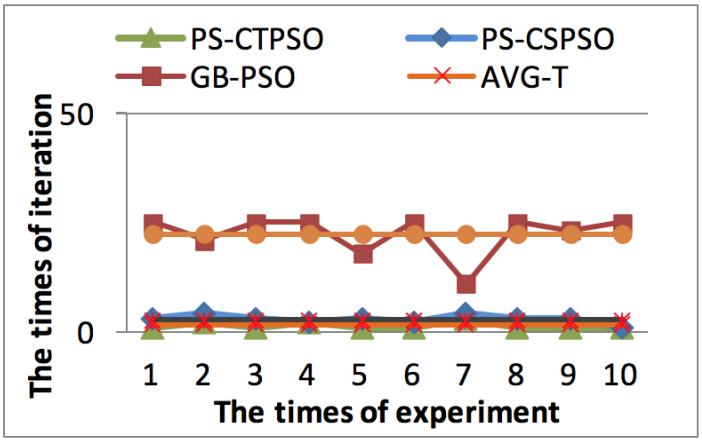
Times Of Iteration			Effective Rate Of Iteration			Whether optimal value		
PS-CTPSO	PS-CSPSO	GB-PSO	PS-CTPSO	PS-CSPSO	GB-PSO	PS-CTPSO	PS-CSPSO	GB-PSO
1	3	25	100%	100%	16%	Yes	Yes	No
2	4	21	100%	75%	38%	Yes	Yes	Yes
1	3	25	100%	100%	24%	Yes	Yes	No
2	2	25	100%	100%	20%	Yes	Yes	No
1	3	18	100%	100%	44%	Yes	Yes	Yes
1	2	25	100%	100%	24%	Yes	Yes	No
3	4	11	66%	75%	55%	Yes	No	Yes
1	3	25	100%	66%	12%	Yes	Yes	No
1	3	23	100%	100%	39%	Yes	Yes	Yes
1	1	25	100%	100%	16%	Yes	Yes	No

Figure 8: The results and comparison of PS-CTPSO and PS-CSPSO, GB-PSO.

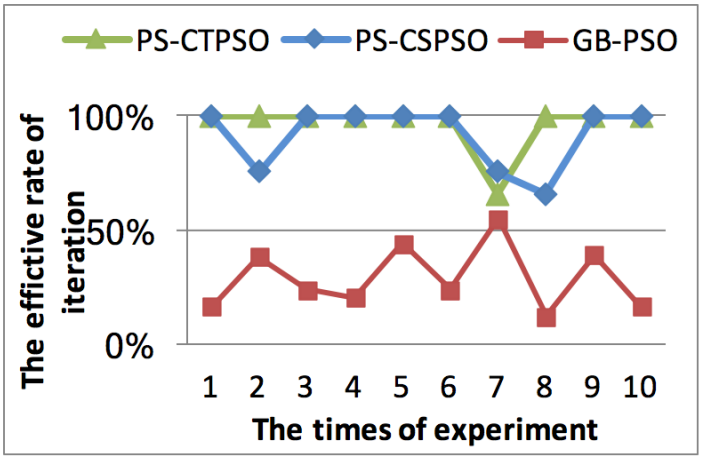
algorithm adopts cotangent sequence to initialise and update particles, whereas the PS-CSPSO algorithm uses a logistic sequence and the GB-PSO algorithm adopts random one. The cotangent sequence exhibits a better performance, as not only it does reduce the iteration times, but also improves the effective rate of iteration. Figure 10 shows the average of effective and optimal rates of iteration, which demonstrates that the performance of searching the optimal solution of PS-CTPSO algorithm is more accurate. As a consequence, the above evaluation supports the fact that PS-CTPSO algorithm effectively addresses local optimal issues.

4.5. Time Consumption

Table 2 shows the time consumption performance.



(a)



(b)

Figure 9: Comparison of iteration times in CS-PSO and PS-CSPSO.

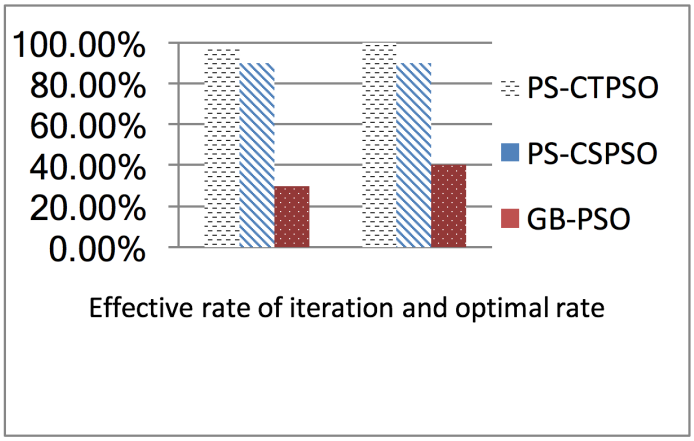


Figure 10: Comparison of effective rate of iteration and optimal rate.

Table 2: Comparison of the ergodicity properties of the three methods.

Date set	PS-CTPSO(ms)	PS-CSPSO(ms)	GB-PSO(ms)
30	< 1	1.8 ± 0.2	2.1 ± 0.5
90	1.1 ± 0.5	4.7 ± 0.3	5.2 ± 1.6
180	2.6 ± 0.7	5.2 ± 0.8	6.7 ± 1.9
450	7.5 ± 1.0	10.6 ± 1.4	19.9 ± 5.7
1350	10.3 ± 1.8	16.8 ± 1.2	30.3 ± 8.7

In comparison with Table 1, the time consumption of the PS-CTPSO algorithm has a considerably smaller value. Figure 11 depicts values of Table 1 showing the trend of the average consumption on each data set. As the number of services is increasing, the amount of time which is saved, is more significant.

This is mainly due to the fact that a feasible service chain in the GB-PSO algorithm may be having invalid services, and these services will affect the performance of searching optimal solution. Even though the impact is not significant when the order of magnitudes of data set is small, invalid services lead to a higher complexity of the optimal solution, especially when addressing large of datasets. Furthermore, in the process of searching optimal solution, GB-PSO

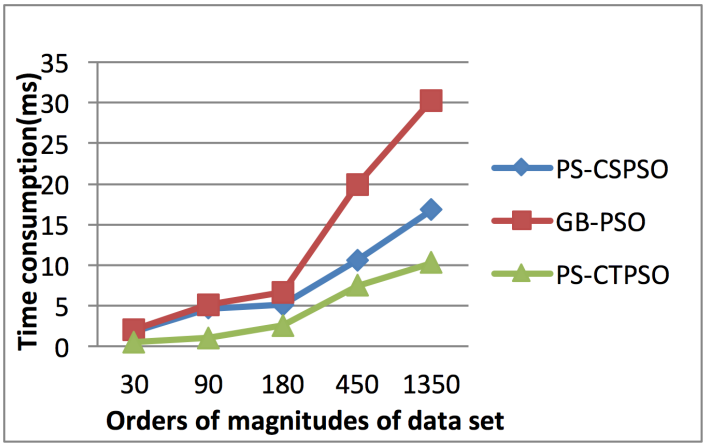


Figure 11: Comparison of time consumption.

algorithm adopts a random searching strategy, which may have a large number of invalid iterations. In fact, an increase in the number of services will lead to more invalid iterations. since PS-CSPSO algorithm adopts a chaotic sequence, it provides a better applicability to Web service combinatorial optimisation problems.

4.6. Diversity in the Combination of the Algorithms

The different combinations are one of the indicators to measure the performance of algorithm. Assume that a user Web service input set requirement is {Address, Date}, and the output set requirement is {City, Weather}. In this data set, there are 6 optimal service chains, respectively defined as

$$\text{Zip Code} \longrightarrow \text{Weather\_i} \longrightarrow \text{Location} \qquad \text{for } i = 0, 1, 2, 3, 4$$

and

$$\text{Zip Code} \longrightarrow \text{BW\_I} \longrightarrow \text{BW\_O} \longrightarrow \text{Location}.$$

Among them the service chain $\text{BW\_I} \longrightarrow \text{BW\_O}$ is equivalent to the service BulportWeatherService_i.

The results are shown in Figure 12, where the PS-CTPSO algorithm attains 5 different optimal service chains totally, containing the service chain

PS-CTPSO	PS-CSPSO	GB-PSO	PS-CTPSO	PS-CSPSO	GB-PSO
Number of service chains			Whether find the replaceable service chain		
5	4	2	No	Yes	Yes

Figure 12: Comparison of diversity of the combination.

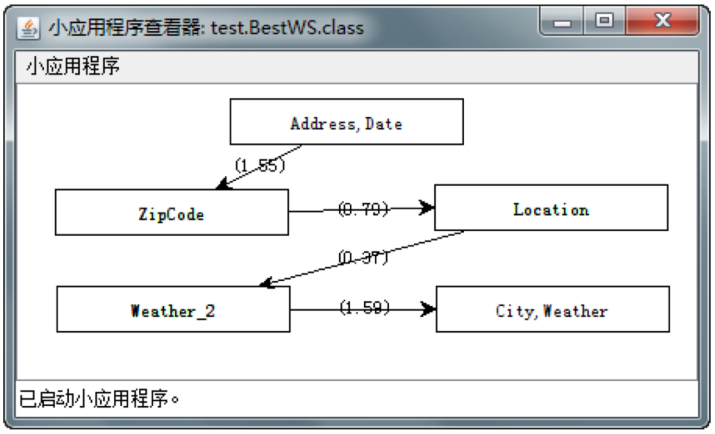


Figure 13: The directed graph of the optimal service chain.

Zip Code $\longrightarrow$ BW_I $\longrightarrow$ BW_O $\longrightarrow$ Location. Furthermore, it identifies 4 different service chains, but it cannot find the replaceable service chain. The GB-PSO algorithm only finds 2 different service chains, due to the GB-PSO algorithm traversing all service nodes, so eventually it must obtain the optimal service chain. However, a large number of service nodes, implies low efficiency. As a consequence, the PS-CTPSO algorithm can ensure the diversity of combination, and using the PS-CTPSO algorithm in BestWS system, the recommended service chain is efficient and reasonable. The directed graph of the optimal service chain is depicted in Figure 13.

4.7. Service Selection and Composition for SaaS

Generally speaking, there are three types of services in Cloud system, including Infrastructure-as-a-Service (IaaS), Platform-as-a-Service(PaaS) and Software-

as-a-Service(SaaS). In particular, SaaS is a software distribution model in which a Cloud provider hosts applications and makes them available to customers over the Internet. As part of the evaluation process, we implemented a tourism planning system, which is a complex service selection and composition application of SaaS, based on the PS-CTPSO algorithm. Assume that a service input set of the tourism planning system is $(Address, Date, Day, Vehicle, Money)$, and the output set requirement is $(City, Weather, TravelDate, ReturnTickets, Tourismplan, Hotel)$. The tourism planning system is composed of multiple optimal service chains, where the output of one service chain may be the input of another one. These include:

$$\begin{aligned}
&(Address, Date) \rightarrow ZipCode \rightarrow Location \rightarrow Weather_2 \rightarrow (TravelDate, City, Weather)\\
&(TravelDate, City, Weather, Days, Money) \rightarrow DomesticAirline \rightarrow Return_Ticket\\
&(Days, Money, Vehicle) \rightarrow ITrip \rightarrow Tourist_Plan\\
&(Days, Money, Hotel_Airbnb) \rightarrow Hotel \rightarrow (Hotel)
\end{aligned} \tag{17}$$

Figure 14 depicts the directed graph of the service selection and composition results, obtained by combining the PS-CTPSO algorithm with the BestWS system.

5. Conclusions

The traditional particle swarm optimisation algorithm tends to be inefficient at identifying a feasible service chain in a large number of Web services. As a consequence, it is likely to spend a considerable time if the random searching strategy is used, which leads to a large number of invalid iterations in the searching process. In this article, we have introduced a predatory search strategy, where a cotangent sequence with chaotic characteristics integrated into the particle swarm optimisation algorithm. More specifically, the predatory search strategy is utilised to search a viable service chain with no invalid service. In the initialisation and update phases, the cotangent sequence replaces a random

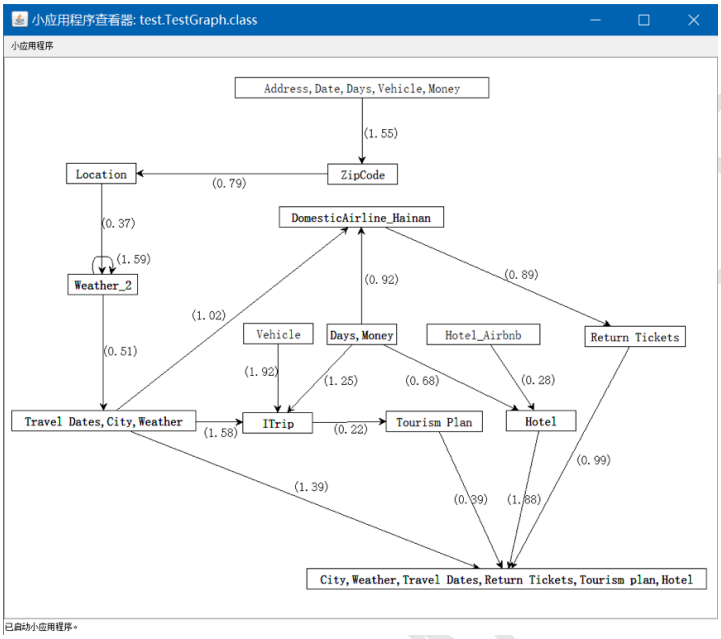


Figure 14: Service selection and composition for SaaS in Cloud.

searching strategy, so improving the global searching ability and efficiency, effectively overcoming the premature problem found frequently in traditional PSO algorithm. Furthermore, logic optimisation ensures the diversity of the final combination scheme.

The validation process demonstrates that the PS-CTPSO algorithm has clear potential in the field of Web service combination optimisation. In the PS-CTPSO algorithm, the combination scheme evaluation function is established, and 4 QoS parameters for evaluating the quality of Web service are considered. However, the personalized requirements of users also need to be considered, such as the lowest execution time and the highest reliability of service. Future research will focus on using the PS-CTPSO algorithm in various Web service servers, to fully assess and implement suitable modifications to further enhance the performance of the algorithm, as well as the valuation function with indi-

vidualised constraints for the combination scheme.

Acknowledgement

This work was jointly sponsored by the National Natural Science Foundation of China under Grants 61472192 and 91646116, the Scientific and Technological Support Project (Society) of Jiangsu Province under Grant BE2016776, the Talent Project in Six Fields of Jiangsu Province under Grant 2015-JNHB-012, and the "333" Scientific Research program of Jiangsu Province under Grant BRA2017228

References

[1] E. Molnar, R. Molnar, N. Kryvinska, M. Gregua, Web intelligence in practice 6 (2014) 149–172.

[2] N. Bashah, N. Kryvinska, D. Thanh, Quality-driven service discovery techniques for open mobile environments and their business applications 4.

[3] G. Horn, K. Jeffery, J. Domaschka, L. Schubert, Analysing the lifecycle of future autonomous cloud applications, in: Proceedings of the 4th International Conference on Cloud Computing and Services Science, CLOSER 2014, SCITEPRESS - Science and Technology Publications, Lda, Portugal, 2014, pp. 569–577. doi:10.5220/0004864705690577.
URL http://dx.doi.org/10.5220/0004864705690577

[4] Y. Wei, M. B. Blake, Service-oriented computing and cloud computing: Challenges and opportunities, IEEE Internet Computing 14 (6) (2010) 72–75. doi:10.1109/MIC.2010.147.

[5] A. Singh, D. Juneja, M. Malhotra, A novel agent based autonomous and service composition framework for cost optimization of resource provisioning in cloud computing, Journal of King Saud University - Computer and Information Sciences 29 (1) (2017) 19 – 28. doi:https://doi.org/10.1016/j.jksuci.2015.09.001.

[6] X. Xu, K. Zhang, D. Li, New chaos-particle swarm optimization algorithm, J. Communications of the ACM 33 (1) 24–30.

[7] J. Kennedy, R. Eberhart, Particle swarm optimization, in: Proceedings of the International Conference on Neural Networks, 1995, pp. 1942–1948.

[8] R.Eberhart, J. Kennedy, A new optimizer using particle swarm theory, in: Proceedings of the International Symposium on Micro Machine and Human Science, 1995, pp. 39–43.

[9] Y. Chang, C. Hsieh, Y. Xu, Introducing the concept of velocity into bare bones particle swarm optimization, in: Proceedings of the International Conference on Information Science, Electronics and Electrical Engineering, 2014, pp. 1580–1584.

[10] K. Lei, A highly efficient particle swarm optimizer for super high-dimensional complex functions optimization, in: Proceedings of the International Conference on Software Engineering and Service Science, 2014, pp. 310–313.

[11] N. Kumari, A. Jha, Frequency control of multi-area power system network using pso based lqr, in: Proceedings of the International Conference of Power, 2014, pp. 1–6.

[12] J. Yao, J. Li, L. Wang, Wireless sensor network localization based on improved particle swarm optimization, in: Proceedings of the International Conference on Computing, Measurement, Control and Sensor Network, 2012, pp. 72–75.

[13] Y. Liao, D. Yau, C. Chen, Evolutionary algorithm to traveling salesman problems, Computers and Mathematics with Applications 33 (64) 788–797.

[14] K. Lee, J. Kimn, Multi-objective particle swarm optimization with preference-based sort and its application to path following footstep optimization for humanoid robots, IEEE Transactions on Evolutionary Computation 17 (64) 755–766.

[15] Y. Chi, F. Sun, W. Wang, An improved particle swarm optimization algorithm with search space zoomed factor and attractor, Chinese Journal of Computers 34 (1) 115–130.

[16] X. Zhao, G. Liu, H. Liu, Particle swarm optimization algorithm based on non-uniform mutation and multiple states perturbation, Chinese Journal of Computers 37 (9) 2058–2070.

[17] S. Chen, L. Ren, F. Xin, Reactive power optimization based on particle swarm optimization and simulated annealing cooperative algorithm, in: Proceedings of the Chinese Control Conference, 2012, pp. 7210–7215.

[18] L. Huang, X. Zhang, X. Huang, A qos optimization for intelligent and dynamic web service composition based on improved pso algorithm, in: Proceedings of the International Conference on Networking and Distributed Computing, 2011, pp. 214–217.

[19] A. Silva, H. Ma, M. Zhang, Graph-based particle swarm optimization approach to qos-aware web service composition and selection, in: Proceedings of the Congress on Evolutionary Computation, 2014, pp. 3128–3134.

[20] P. Zhang, Z. Jing, Z. Zhang, Dynamic web service com-position based on discrete particle swarm optimization, Journal of Computer Science 42 (6) 71–75.

[21] T. Wen, G. Sheng, Q. Guo, Web service composition based on modified particle swarm optimization, Chinese Journal of Computers 36 (5) 1031–1046.

[22] F. Yong-Yi, S. Yang, J. Kuo, Search based approach to forecasting qos attributes of web services using genetic programming, Information and Software Technology 80 (5) 158–174.

[23] A. Linhares, Preying on optima: A predatory search strategy for combinatorial problems, in: Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics, 1998, pp. 2974–29784.

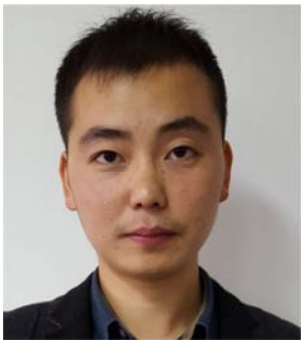
[24] N. Dong, H. Li, X. Liu, Chaotic particle swarm optimization algorithm parametric identification of bouc-wen hysteresis model for piezoelectric ceramic actuator, in: Proceedings of the Conference on Chinese Control and Decision, 2013, pp. 2435–2440.

[25] X. Xu, H. Rong, M. Trovati, Cs-pso: chaotic particle swarm optimization algorithm for solving combinatorial optimization problems, Soft Computing 60 (1) 1–13.

Algorithm 3 Discovering a relevant services for composition

Input: $N, M, K[], C[], W[][], Q[][]$
Output: G
$Best_F = v$ {v is the theoretical best fitness function value}
for $i = 0$ to $N - 1$ **do**
  $iX[i] = V[i] = P[i] = Initialize(C, W)$ {chaos initialisation}
  $F[i] = ComputerFitness(X[i], Q[i])$ {calculate the fitness value}
end for
$index = Get_Global_Best(F[i])$ {obtain the index of the global best particle}
$F^g = F[index]$
$G = X[index]$
while $F^g != Best_F \&\& Iterations != MaxCount$ **do**
  for $i = 0$ to $N - 1$ **do**
    $V[i] = UpdateSpeed(X[i], V[i], P[i], G)$ {update speed of each particle}
    $X[i] = UpdatePos(X[i], P[i], G, V[i], C, W)$ {update position of each particle}
    $= ComputerFitness(X[i], Q[i])$
  end for
  $index = Get_Global_Best(F[i])$
  $F^g = F[index]$
  $G = X[index]$
end while
return G

Xiaolong Xu is a professor in the School of Computer Science, Nanjing University of Posts & Telecommunications, Nanjing, China. He received his BSc in computer and its applications, MSc in computer software and theories and PhD in communications and information systems, in 1999, 2002 and 2008, respectively. He is a senior member of China Computer Federation. He conducts research in areas of Cloud Computing, Big Data, Information Security and Novel Network Computing Technologies. He has successfully led and completed a number of projects, including projects sponsored by the National Science Fund of China. He has published more than 100 Journal and conference papers and 5 books. He is authorised 38 patents by the State Intellectual Property Office of China as the first inventor. He was rated as excellent young professor of Jiangsu Province in 2014, selected as the high-level creative talents of Jiangsu province in 2015, and won the title of outstanding expert in the area of computer science and technology.



Hanzhong Rong is a postgraduate student majored in software engineering at Nanjing University of Posts and Telecommunications, China. He received his B.S. degree in computer science and technology at Nanjing University of Posts and Telecommunications, China, in 2014. His main research interests include Web Services and Cloud Computing.

Ella Pereira (corresponding author) is Professor of Computing in the Department of Computer Science, Edge Hill University, UK. Ella holds PhD in Distributed Software Development from Liverpool John Moores University, UK and MSc in Computer Systems and BSc in Mechanical Engineering from Georgian Technical University, Georgia. She is a Fellow of HEA and member of BCS. Ella's main research interests and expertise are on cloud computing, IoT and mobile application development. She has growing interest and number of collaborative projects in utilising advanced computing technology for healthcare. Ella has published widely, and won best paper and research leadership awards. She organises and chairs various international conferences and workshops, has edited books and conference proceedings, and acted as editor in chief for academic journals.

Marcello Trovati is a reader at the department of Computer Science, Edge Hill University, UK. After having obtained his PhD in Mathematics at the University of Exeter in 2007, Marcello has worked in industry, including at Dublin IBM Research Lab carrying out research mainly in the field of knowledge discovery, text mining, and mathematical modelling. Later he held a number of academic posts at different UK universities before joining Edge Hill University in 2016. Marcello is involved in several research themes and projects. He is co-leading the STEM Data Research centre and is actively involved in the Productivity and Innovation Lab, aiming to collaborate and support SMEs in Lancashire.

Marcello's main interests include: Mathematical Modelling, Data Science, Big Data Analytics, Network Theory, Machine Learning, Data and Text Mining.

Highlights

- A particle swarm optimisation algorithm for web service combination proposed.
- The PS-CTPSO algorithm is based on the predatory search strategy.
- This strategy is utilised to search a viable service chain with no invalid service.
- For web service combination the cotangent sequence is more suitable than chaotic one.
- Logic optimisation ensures the diversity of the final combination scheme.